Software Design, Modelling and Analysis in UML

Lecture 03: Object Constraint Language

2016-10-27

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

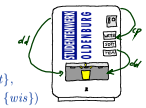
Content

(Core) OCL Syntax OMG (2006)

Overview

$expr ::=$	w	$: \tau(w)$
	$\mid expr_1 =_\tau expr_2$	$: \tau \times \tau \to Bool$
	$\mid \mathsf{oclIsUndefined}_\tau(expr_1)$	$: \tau \to Bool$
	$\mid \{expr_1, \ldots, expr_n\}$	$: \tau \times \cdots \times \tau \to Set(\tau)$
	$\mid \mathsf{size}(expr_1)$	$: Set(\tau) \to Int$
	$\mid \mathsf{allInstances}_C$	$: Set(\tau_C)$
	$\mid v(expr_1)$	$: \tau_C \to \tau(v)$
	$\mid r_1(expr_1)$	$: \tau_C \to \tau_D$
	$\mid r_2(expr_1)$	$: \tau_C \to Set(\tau_D)$
	$\mid \mathsf{true}, \mathsf{false}$	$: Bool$
	$\mid \mathsf{not}\ expr_1$	$: Bool \to Bool$
	$\mid expr_1\ \{\mathsf{and}, \mathsf{or}, \mathsf{implies}\}\ expr_2$	$: Bool \times Bool \to Bool$
	$\mid \ldots$	
	$\mid \mathsf{OclUndefined}_\tau$	$: \tau$
	$\mid expr_1\text{->}\mathsf{iterate}(w_1 : T_1 ; w_2 : T_2 = expr_2 \mid expr_3)$	$: Set(\tau_0) \to \tau_{T_2}$
$context ::=$	$\mathsf{context}\ w_1 : T_1, \ldots, w_n : T_n\ \mathsf{inv} : expr$	$: Bool$

Recall: Vending Machine Structure



$$\mathscr{S} = (\{Bool, Int\}, \{VM, CP, DD\},$$
$$\{cp : CP_*, dd : DD_{0,1}, wen : Bool, wis : Nat\},$$
$$\{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$

Claim: this is a proper OCL constraint over $\mathscr{S}$:

$$\mathsf{context}\ \ CP\ \ \mathsf{inv}:\ \ wen\ \ \mathsf{implies}\ \ dd\,.\,wis > 0$$

Plan

$expr ::=$	w	$: \tau(w)$
7/4	$\mid expr_1 =_\tau expr_2$	$: \tau \times \tau \to Bool$
	$\mid \mathsf{oclIsUndefined}_\tau(expr_1)$	$: \tau \to Bool$
	$\mid \{expr_1, \ldots, expr_n\}$	$: \tau \times \cdots \times \tau \to Set(\tau)$
	$\mid \mathsf{size}(expr_1)$	$: Set(\tau) \to Int$
	$\mid \mathsf{allInstances}_C$	$: Set(\tau_C)$
	$\mid v(expr_1)$	$: \tau_C \to \tau(v)$
	$\mid r_1(expr_1)$	$: \tau_C \to \tau_D$
	$\mid r_2(expr_1)$	$: \tau_C \to Set(\tau_D)$
2/4	$\mid \mathsf{true}, \mathsf{false}$	$: Bool$
	$\mid \mathsf{not}\ expr_1$	$: Bool \to Bool$
	$\mid expr_1\ \{\mathsf{and}, \mathsf{or}, \mathsf{implies}\}\ expr_2$	$: Bool \times Bool \to Bool$
	$\mid \ldots$	
	$\mid \mathsf{OclUndefined}_\tau$	$: \tau$
3/4	$\mid expr_1\text{->}\mathsf{iterate}(w_1 : T_1 ; w_2 : T_2 = expr_2 \mid expr_3)$	$: Set(\tau_0) \to \tau_{T_2}$
4/4	$context ::= \mathsf{context}\ w_1 : T_1, \ldots, w_n : T_n\ \mathsf{inv} : expr$	$: Bool$

OCL Syntax 1/4: Expressions

$expr ::=$

w	$: \tau(w)$
$\mid expr_1 =_\tau expr_2$	$: \tau \times \tau \to Bool$
$\mid \text{oclIsUndefined}_\tau(expr_1)$	$: \tau \to Bool$
$\mid \{expr_1, \ldots, expr_n\}$	$: \tau \times \cdots \times \tau \to Set(\tau)$
$\mid \text{isEmpty}(expr_1)$	$: Set(\tau) \to Bool$
$\mid \text{size}(expr_1)$	$: Set(\tau) \to Int$
$\mid \text{allInstances}_C$	$: Set(\tau_C)$
$\mid v(expr_1)$	$: \tau_C \to \tau(v)$
$\mid r_1(expr_1)$	$: \tau_C \to \tau_D$
$\mid r_2(expr_1)$	$: \tau_C \to Set(\tau_D)$

where $v : \tau \in atr(C)$, $\tau \in \mathscr{T}$,
where $r_1 : D_{0,1} \in atr(C), C, D \in \mathscr{C}$,
where $r_2 : D_* \in atr(C), C, D \in \mathscr{C}$.

Where, given $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$,

- $w \in W \supseteq \{self_C : \tau_C \mid C \in \mathscr{C}\}$ is a set of typed logical variables, w has type $\tau(w)$
- τ is any type from $\mathscr{T} \cup T_B \cup T_{\mathscr{C}} \cup \{Set(\tau_0) \mid \tau_0 \in \mathscr{T} \cup T_B \cup T_{\mathscr{C}}\}$
  - T_B is a set of (OCL) basic types, in the following we use $T_B = \{Bool, Int, String\}$
  - $T_{\mathscr{C}} = \{\tau_C \mid C \in \mathscr{C}\}$ is the set of object types.
  - $Set(\tau_0)$ denotes the set-of-τ_0 type for $\tau_0 \in T_B \cup T_{\mathscr{C}}$ (sufficient because of "flattening" (cf. standard)).

Expression Examples

$expr ::=$

w	$: \tau(w)$	$\mid \text{size}(expr_1)$	$: Set(\tau) \to Int$
$\mid expr_1 =_\tau expr_2$	$: \tau \times \tau \to Bool$	$\mid \text{allInstances}_C$	$: Set(\tau_C)$
$\mid \text{oclIsUndefined}_\tau(expr_1)$	$: \tau \to Bool$	$\mid v(expr_1)$	$: \tau_C \to \tau(v)$
$\mid \{expr_1, \ldots, expr_n\}$	$: \tau \times \cdots \times \tau \to Set(\tau)$	$\mid r_1(expr_1)$	$: \tau_C \to \tau_D$
$\mid \text{isEmpty}(expr_1)$	$: Set(\tau) \to Bool$	$\mid r_2(expr_1)$	$: \tau_C \to Set(\tau_D)$

$\mathscr{S}_0 = (\{Int_b\}, \{\bar{C}, \bar{D}\}, \{x : Int_b, p : \bar{C}_{0,1}, n : \bar{C}_*\}, \{\bar{C} \mapsto \{p, n\}, \bar{D} \mapsto \{x\}\})$

Expression Examples

$expr ::=$

w	$: \tau(w)$	$\mid \text{size}(expr_1)$	$: Set(\tau) \to Int$
$\mid expr_1 =_\tau expr_2$	$: \tau \times \tau \to Bool$	$\mid \text{allInstances}_C$	$: Set(\tau_C)$
$\mid \text{oclIsUndefined}_\tau(expr_1)$	$: \tau \to Bool$	$\mid v(expr_1)$	$: \tau_C \to \tau(v)$
$\mid \{expr_1, \ldots, expr_n\}$	$: \tau \times \cdots \times \tau \to Set(\tau)$	$\mid r_1(expr_1)$	$: \tau_C \to \tau_D$
$\mid \text{isEmpty}(expr_1)$	$: Set(\tau) \to Bool$	$\mid r_2(expr_1)$	$: \tau_C \to Set(\tau_D)$

$\mathscr{S}_0 = (\{Int\}, \{C, D\}, \{x : Int, p : C_{0,1}, n : C_*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$

context CP inv : wen implies $dd.wis > 0$

Notational Conventions for Expressions

- Each expression

$$\omega(expr_1, expr_2, \ldots, expr_n) : \tau_1 \times \cdots \times \tau_n \to \tau$$

may alternatively be written ("abbreviated as")

- $expr_1 \;.\; \omega(expr_2, \ldots, expr_n)$ if τ_1 is an object type, i.e. if $\tau_1 \in T_{\mathscr{C}}$.
- $expr_1 \;\texttt{->}\; \omega(expr_2, \ldots, expr_n)$ if τ_1 is a collection type (here: only sets), i.e. if $\tau_1 = Set(\tau_0)$ for some $\tau_0 \in T_B \cup T_{\mathscr{C}}$.

- Examples: $(\{Int\}, \{C, D\}, \{x : Int, p : C_{0,1}, n : C_*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$
- $self_C.p$
- $self_C.p.n$
- $self_C.p.n$ -> isEmpty
- context CP inv : wen implies $dd.wis > 0$ $\quad (atr(CP) = \{wen : Bool, dd : DD_{0,1}\})$

OCL Syntax 2/4: Constants & Arithmetics

For example:

$expr ::=$ ...

$\mid$ true, false	$: Bool$
$\mid expr_1$ {and, or, implies} $expr_2$	$: Bool \times Bool \to Bool$
$\mid$ not $expr_1$	$: Bool \to Bool$
$\mid 0, -1, 1, -2, 2, \ldots$	$: Int$
$\mid expr_1$ $\{+, -, \ldots\}$ $expr_2$	$: Int \times Int \to Int$
$\mid expr_1$ $\{<, \leq, \ldots\}$ $expr_2$	$: Int \times Int \to Bool$
$\mid \text{OclUndefined}_\tau$	$: \tau$

Generalised notation: (prefix normal form)

$expr ::= \omega(expr_1, \ldots, expr_n) \quad : \tau_1 \times \cdots \times \tau_n \to \tau$

with $\omega \in \{+, -, \ldots\}$

Constants & Arithmetics Examples

$expr ::=$ ...

$\mid$ true, false	$: Bool$
$\mid expr_1$ {and, or, implies} $expr_2$	$: Bool \times Bool \to Bool$
$\mid$ not $expr_1$	$: Bool \to Bool$
$\mid 0, -1, 1, -2, 2, \ldots$	$: Int$
$\mid expr_1$ $\{+, -, \ldots\}$ $expr_2$	$: Int \times Int \to Int$
$\mid expr_1$ $\{<, \leq, \ldots\}$ $expr_2$	$: Int \times Int \to Bool$
$\mid \text{OclUndefined}_\tau$	$: \tau$

$\mathscr{S}_0 = (\{Int\}, \{C, D\}, \{x : Int, p : C_{0,1}, n : C_*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$

context CP inv : wen implies $dd.wis > 0$

OCL Syntax 3/4: Iterate

$$expr ::= \cdots \mid expr_1\text{->iterate}(w_1 : T_1 ; w_2 : T_2 = expr_2 \mid expr_3)$$

or, with a little renaming,

$$expr ::= \cdots \mid expr_1\text{->iterate}(iter : T_1; result : T_2 = expr_2 \mid expr_3)$$

where

- $expr_1$ is of a collection type (here: a set $Set(\tau_0)$ for some τ_0).
- $iter \in W$ is called iterator, of the type denoted by T_1
  (if T_1 is omitted, τ_0 is assumed as type of $iter$)
- $result \in W$ is called result variable, gets type τ_2 denoted by T_2,
- $expr_2$ in an expression of type τ_2 giving the initial value for $result$,
  ($\mathsf{OclUndefined}_{\tau_2}$, if omitted)
- $expr_3$ is an expression of type τ_2,
  in particular $iter$ and $result$ may appear in $expr_3$.

Iterate Example

$$\mathscr{S} = (\{Bool, \overset{Nat}{Int}\}, \{VM, CP, DD\},$$
$$\{cp : CP_*, dd : DD_{0,1}, wen : Bool, wis : Nat\},$$
$$\{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$

$$expr ::= expr_1\text{->iterate}(w_1 : T_1; w_2 : T_2 = expr_2 \mid expr_3)$$

- $\underbrace{cp\,(self_{VM})}_{:Set(\tau_{CP})}$ ->iterate($\underbrace{iter}_{:\tau_{CP}}$; res : Int = 0 | res + $\underbrace{iter.dd.wis}_{:Nat \in \mathcal{T}_{\mathcal{S}}}$)
- cp ($self_{VM}$) ->iterate(iter ; res : Bool = true | res and iter.wen)

Abbreviations on Top of Iterate

$$expr ::= expr_1\text{->iterate}(w_1 : T_1; w_2 : T_2 = expr_2 \mid expr_3)$$

- $expr_1$->forAll($w_1 : \overset{\blacksquare}{T_1} \mid expr_3$) ("$\forall w_1 \in expr_1 \bullet expr_3$")
  is an abbreviation for
  $$expr_1\text{->iterate}(w_1 : T_1; w_2 : Bool = \mathsf{true} \mid w_2 \text{ and } expr_3).$$
- $expr_1$->Exists($w : \overset{\blacksquare}{T_1} \mid expr_3$)
  is an abbreviation for
  $$expr_1\text{->iterate}(w : T_1 ; w_2 : Bool = false \mid w_2 \text{ or } expr_3)$$

To ensure confusion, we may again omit all kinds of things, cf. OMG (2006).

Recall: Overview

$expr ::=$	w	$: \tau(w)$
	$\mid expr_1 =_\tau expr_2$	$: \tau \times \tau \to Bool$
	$\mid \mathsf{oclIsUndefined}_\tau(expr_1)$	$: \tau \to Bool$
	$\mid \{expr_1, \ldots expr_n\}$	$: \tau \times \cdots \times \tau \to Set(\tau)$
	$\mid \mathsf{size}(expr_1)$	$: Set(\tau) \to Int$
	$\mid \mathsf{allInstances}_C$	$: Set(\tau_C)$
	$\mid v(expr_1)$	$: \tau_C \to \tau(w)$
	$\mid r_1(expr_1)$	$: \tau_C \to \tau_D$
	$\mid r_2(expr_1)$	$: \tau_C \to Set(\tau_D)$
	$\mid \mathsf{true}, \mathsf{false}$	$: Bool$
	$\mid \mathsf{not}\ expr_1$	$: Bool \to Bool$
	$\mid expr_1\ \{\mathsf{and}, \mathsf{or}, \mathsf{implies}\}\ expr_2$	$: Bool \times Bool \to Bool$
	$\mid \ldots$	
	$\mid \mathsf{OclUndefined}_\tau$	$: \tau$
	$\mid expr_1\text{->iterate}(w_1 : T_1 ; w_2 : T_2 = expr_2 \mid expr_3)$	$: Set(\tau_0) \to \tau_{T_2}$
$context ::=$	$\mathsf{context}\ w_1 : T_{1,} \ldots, w_n : T_n\ \mathsf{inv} : expr$	$: Bool$

More Iterate Examples

$$\mathscr{S} = (\{Bool, \overset{Nat}{Int}\}, \{VM, CP, DD\},$$
$$\{cp : CP_*, dd : DD_{0,1}, wen : Bool, wis : Nat\},$$
$$\{VM \mapsto \{cp, dd\}, CP \mapsto \{wen, dd\}, DD \mapsto \{wis\}\})$$

$$expr ::= expr_1\text{->iterate}(w_1 : T_1; w_2 : T_2 = expr_2 \mid expr_3)$$

allInstances$_{CP}$ ->iterate($self_{CP}$: CP ; res : Bool = true |
res and ($self_{CP}$.wen implies $self_{CP}$.dd.wis > 0))

or:

allInstances$_{CP}$ ->forAll($self_{CP}$ | $self_{CP}$.wen implies $self_{CP}$.dd.wis > 0)

context CP inv : wen implies dd.wis > 0

OCL Syntax 4/4: Context

Syntax: (Assuming signature $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$.)

$$context ::= \mathsf{context}\ w_1 : T_{1,} \ldots, w_n : T_n\ \mathsf{inv} : expr$$

where $T_i \in \mathscr{C}$ and $w_i : \tau_{T_i} \in W$ for all $1 \le i \le n$, $n \ge 0$.

Semantics:

$$\mathsf{context}\ w_1 : C_1, \ldots, w_n : C_n\ \mathsf{inv} : expr$$

is (just) an **abbreviation** for

$$\begin{array}{l}\mathsf{allInstances}_{C_1}\text{->forAll}(w_1 : \overset{\blacksquare}{C_1} \mid \\ \quad \ldots \\ \quad\quad \mathsf{allInstances}_{C_n}\text{->forAll}(w_n : \overset{\blacksquare}{C_n} \mid \\ \quad\quad\quad expr \\ \quad\quad) \\ \quad \ldots \\ )\end{array}$$

Context: More Notational Conventions

- For

  context $self : T$ inv : $expr$

  we may **alternatively** write ("**abbreviate as**")

  context T inv : $expr$

- **Within** the latter abbreviation, we may omit the "$self$" in expression $expr$, i.e. for

  context T inv : $self.v$

  (which is an abbreviation for context T inv : $v(self)$)

  we may alternatively write ("**abbreviate as**")

  context T inv : v

The Running Example

context CP inv : wen implies $dd.wis > 0$

context self : CP inv : wen implies dd.wis > 0

context self : CP inv : self.wen implies self.dd.wis > 0

allInstances_CP ->forAll(self | self.wen implies self.dd.wis > 0)

-- " -- iterate(....)

Recall: Overview

$expr ::=$	w	$: \tau(w)$
	$\mid expr_1 =_\tau expr_2$	$: \tau \times \tau \to Bool$
	$\mid$ oclIsUndefined$_\tau(expr_1)$	$: \tau \to Bool$
	$\mid \{expr_1, \ldots, expr_n\}$	$: \tau \times \cdots \times \tau \to Set(\tau)$
	$\mid$ size$(expr_1)$	$: Set(\tau) \to Int$
	$\mid$ allInstances$_C$	$: Set(\tau_C)$
	$\mid v(expr_1)$	$: \tau_C \to \tau(v)$
	$\mid r_1(expr_1)$	$: \tau_C \to \tau_D$
	$\mid r_2(expr_1)$	$: \tau_C \to Set(\tau_D)$
	$\mid$ true, false	$: Bool$
	$\mid$ not $expr_1$	$: Bool \to Bool$
	$\mid expr_1$ {and, or, implies} $expr_2$	$: Bool \times Bool \to Bool$
	$\mid \ldots$	
	$\mid$ OclUndefined$_\tau$	$: \tau$
	$\mid expr_1$->iterate$(w_1 : T_1 ; w_2 : T_2 = expr_2 \mid expr_3)$	$: Set(\tau_0) \to \tau_{T_2}$
$context ::=$	context $w_1 : T_1, \ldots, w_n : T_n$ inv : $expr$	$: Bool$

"Not Interesting"

Among others:

- Enumeration types
- Type hierarchy
- Complete list of arithmetical operators
- The two other collection types Bag and Sequence
- Casting
- Runtime type information
- Pre/post conditions
  (maybe later, when we officially know what an operation is)
- ... context f pre: expr1 post: expr2

References

References

OMG (2006). Object Constraint Language, version 2.0. Technical Report formal/06-05-01.

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.

Warmer, J. and Kleppe, A. (1999). *The Object Constraint Language*. Addison-Wesley.